

Empirical Evaluation of the Heat-Diffusion Collection Protocol for Wireless Sensor Networks

Pradipta Ghosh^{a,*}, He Ren^b, Reza Banirazi^a, Bhaskar Krishnamachari^a,
Edmond Jonckheere^a

^a*Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, California, USA*

^b*Electrical Engineering, Stanford University, Stanford, California, USA*

Abstract

Heat-Diffusion (HD) routing is our recently-developed queue-aware routing policy for multi-hop wireless networks inspired by Thermodynamics. In the prior theoretical studies, we have shown that HD routing guarantees throughput optimality, minimizes a quadratic routing cost, minimizes queue congestion on the network, and provides a trade-off between routing cost and queueing delay that is Pareto-Optimal. While striking, these guarantees are based on idealized assumptions (including global synchronization, centralized control, and infinite buffers) and heretofore have only been evaluated through simplified numerical simulations. We present here the first practical decentralized Heat-Diffusion Collection Protocol (HDCP) for wireless sensor networks and detail its implementation on Contiki OS. We present a thorough evaluation of HDCP based on real testbed experiments, including a comparative analysis of its performance with respect to the state of the art Collection Tree Protocol (CTP) and Back-pressure Collection Protocol (BCP) for wireless sensor networks. We find that HDCP has a significantly higher throughput region and greater resilience to interference compared to CTP. However, we also find that the best performance of HDCP is comparable to the best performance of BCP, due to the similarity

*Corresponding author

Email addresses: pradiptg@usc.edu (Pradipta Ghosh), rhilogin@gmail.com (He Ren), banirazi@gmail.com (Reza Banirazi), bkrishna@usc.edu (Bhaskar Krishnamachari), jonckhee@usc.edu (Edmond Jonckheere)

in their neighbor rankings, which we verify through a Kendall's-Tau test.

1. Introduction

Low power wireless sensor networks tend to be used for low-data rate applications. However, their scaling in terms of network size as well as operation under low duty cycles is often limited due to bandwidth constraints. Routing algorithms that can utilize the full bandwidth capacity of the network are therefore very important and continue to be a subject of research and development.

In network theory, the ability to fully utilize the available bandwidth in a network is tied to the notion of throughput optimality. An algorithm is said to be throughput optimal if it has the ability to maintain stable queues at any set of arrival rates that could possibly be stabilized by at least one algorithm. The Back-Pressure (BP) routing algorithm [1] was the first queue-aware routing protocol to offer in theory a throughput optimality guarantee under general channel state and traffic conditions. It has been translated to practice in the form of the Backpressure Collection Protocol [2] for wireless sensor networks, which was shown to provide improved capacity and robustness to channel dynamics compared to the state of the art queue-unaware tree-based routing protocols.

The Heat Diffusion (HD) algorithm [3] is our recently proposed alternate queue-aware throughput optimal routing policy for wireless networks. It is derived from a combinatorial analog of the classical Heat Diffusion equation in Thermodynamics (where queue size is analogous to temperature, and packet flow to heat flow) that takes into account wireless interference constraints. Moreover, in [4] we have shown that as the underlying mathematical formalism is also essentially the same as current flows in resistive circuits, link penalties corresponding to resistances can be incorporated into HD routing in a way that allows for minimizing a specific form of average routing cost referred to as the Dirichlet routing cost. The Dirichlet routing cost is defined as the product of a link's cost and square of the respective link's flow rate. Moreover, the HD algorithm also minimizes the overall queue congestion of the network among

the class of throughput optimal algorithms that make decision based on only current queue occupancies and channel statistics. The HD routing algorithm guarantees to operate on the Pareto boundary if both routing costs and queue occupancies are considered in the objective function.

Thus, in theory, the HD routing algorithm goes beyond just throughput optimality guarantees to provide additional significant improvements in average queue sizes (delay) and average routing costs (such as ETX) compared to traditional Backpressure routing. However, to date, this HD algorithm has remained a theoretical, idealized construct that requires a centralized implementation based on a complete knowledge of a network and a NP-hard scheduling procedure at each time, and assumes that buffer sizes are unbounded at all nodes. What has been missing in the literature is a practical implementation of the HD policy that is distributed and works with finite buffer lengths, and whose performance is studied comprehensively on a real wireless testbed. We seek to address this gap.

1.1. Our Contribution

Our contribution in this paper is multi-fold. **First**, we present the first-ever decentralized version of the Heat-Diffusion algorithm and detail a Contiki OS [5] based practical protocol implementation for data collection in wireless sensor networks: the Heat Diffusion Collection Protocol (HDCP).

Second, we propose practically-motivated enhancements of the original Heat Diffusion algorithm in our HDCP protocol implementation, including modifications to the link weight calculations, and a link switching scheme to diversify the link usage.

Third, we propose and evaluate a new method of dynamic ETX calculation suitable for any dynamic routing algorithm, including the previously proposed Backpressure Collection Protocol (BCP) [2] as well as HDCP.

Fourth, we present and analyze the data collected from an extensive set of practical experiments conducted with HDCP utilizing forty five nodes on a real wireless sensor network testbed. Based on these data, we discuss the variation

in the performance of HDCP under different parameters.

Fifth, we compare HDCP with a Contiki-OS implementation of the Backpressure Collection Protocol (BCP) [2] as well as the well known Collection Tree Protocol (CTP) [6]. We show that on the real testbed, HDCP offers significant improvements in performance over CTP in terms of throughput as well as resilience to external interference. We also show that the performance of HDCP is similar to BCP, and through evaluation of a Kendall’s Tau similarity measure, show that this is due to similar rankings among the neighbors.

Finally, we also verify that HDCP performs well with a low power communication stack (CX-MAC, a version of X-MAC[7] that is provided in Contiki, with 5% duty cycle).

2. Related Work

Besides the original Backpressure routing algorithm, other throughput optimal policies [8, 9, 10] have also been proposed in the existing network theory literature. The HD algorithm also provides the same throughput optimality guarantee in theory. However, what motivated us to implement HD were the striking additional expected performance capabilities (based on our theoretical results)—that it also offers a Pareto-optimal trade-off between routing cost and queue congestion.

There have also been several reductions of Backpressure routing to practice in the form of distributed protocols, pragmatically implemented and empirically evaluated for different types of wireless networks [2, 11, 12]. Most relevant to the present work is the Backpressure Collection Protocol (BCP) developed by Moeller *et al.*, the first ever implementation of dynamic queue-aware routing in wireless sensor networks [2]. Our present work is informed by the BCP approach to implement Backpressure routing in a distributed manner and we also directly compare the performance of the new HDCP protocol with BCP.

Besides BCP, there are a number of other prior works on routing and collection protocols for wireless sensor networks, including the Collection Tree Protocol (CTP) [6], Glossy [13], Dozer [14], Low-power Wireless Bus [15], ORW [16]

and Oppcast [17]. We provide a side by side comparison of HDCP with the well-known CTP and BCP protocols. We believe this provides a meaningful comparison with a state of the art minimum cost quasi-static routing protocol as well as a state of the art queue and cost-aware dynamic routing protocol.

In recent years there has been a significant focus in developing networking protocols that are IP-friendly, such as RPL [18]. While the present paper does not focus on providing an IP-compliant version of HD, there is prior work on extending BCP to handle IP packets [19] and we believe that a similar approach could be adopted to enable IP operation for HDCP in the future.

In our prior works, we have presented the idealized Heat Diffusion routing algorithm [3, 4]. All of these are network theory papers that spell out a centralized algorithm, assume global synchronization, assume that at each time step a NP-hard Maximum Weight Independent Set problem can be solved, and that all queues are of unlimited size, and under these assumptions prove various properties of the HD algorithm. The only evaluations presented in these works are idealized MATLAB simulations. This work is clearly inspired by and built up on our earlier works on HD routing, but is the first to develop and implement it as a realistic distributed protocol (HDCP) and evaluate it on a real testbed.

3. Preliminaries and Background

The general idea behind dynamic queue-aware routing algorithms such as the Backpressure [1] and the Heat Diffusion [3] is that they do not require any explicit path computation. Instead, the next-hop for each packet depends on queue-differential weights that are functions of the local queue occupancy information and link state information at each node. Next, we briefly discuss the Backpressure routing algorithm, first proposed by Tassiulas and Ephremides [1], and extended by Neely *et al.* [20, 21] and then give more concrete details on the Heat Diffusion routing algorithm proposed by Banirazi *et al* [3].

3.1. Backpressure Routing

The original Backpressure (BP) routing algorithm [1] consists of three major steps: BP weighing, BP scheduling and BP forwarding. The BP algorithm

uses the information about estimated channel capacities $\mu_{ij}(n)$ and the queue backlogs $q_i(n)$ to make the routing decisions at each time slot n . This follows a brief description of the BP routing steps including the penalty optimization extension introduced by Neely *et al.*.

3.1.1. BP Weighing

For each link ij in the network, find the queue differential, $q_{ij}(n) = q_i(n) - q_j(n)$. Next, assign some weights to the links based on the queue differential as follows.

$$w_{ij}(n) = \mu_{ij}(n)q_{ij}(n) \quad (1)$$

In the original BP, only the queue stabilities are considered. To incorporate the routing cost into the BP, the drift-plus-penalty approach [20, 21] was proposed, which we refer to as the V-parameter BP algorithm. In this approach, a route usage cost is added as a negative penalty in the weight calculation as follows.

$$w_{ij}(n) = \mu_{ij}(n)(q_{ij}(n) - V\theta_{ij}) \quad (2)$$

where $V \in [0, \infty)$ determines the importance of the link penalty and θ_{ij} is the link penalty which depends on the link utility or cost function along with some penalty functions.

3.1.2. BP Scheduling

Find or choose a scheduling vector $\pi \in \Pi$ that maximizes the sum of the weights of the activated links. In other words, choose a scheduling vector π such that

$$\phi(n) = \arg \max_{\pi \in \Pi} \sum_{ij \in \mathcal{E}} \pi_{ij} w_{ij}(n) \quad (3)$$

In case of ties, the scheduling vector is chosen randomly from the solution set.

3.1.3. BP Forwarding

Based on the scheduling vector from step(2), if a link is active at time-slot n i.e., $\pi_{ij}(n) = 1$, and if the link weight $w_{ij}(n) > 0$, then transmit packets on

that link at full capacity $\mu_{ij}(n)$. Null packets are sent if a node doesn't have enough packets to send.

3.2. Heat Diffusion Routing

Heat Diffusion (HD) [3] routing has been derived from the combinatorial analogue of classical Heat-Diffusion equation in Thermodynamics. Similar to the BP routing, in the HD routing, the nodes make routing decisions to put f_{ij} packets over a link in a discrete time, based on queue differentials $q_{ij}(n)$, link capacities $\mu_{ij}(n)$, and channel cost factor $\rho_{ij}(n)$.

The optimization goal of the HD algorithm for a wireless network in theory can be described as follows [3, 4]:

$$\begin{aligned} &\text{Minimize: } (1 - \beta)\overline{Q} + \beta\overline{R} \\ &\text{Subject to: (1) Throughput optimality.} \\ &\quad \quad \quad (2) \text{ Network constraints.} \end{aligned} \tag{4}$$

where $\overline{R} = \sum_{ij \in \mathcal{E}} \overline{\rho_{ij}(f_{ij})^2}$ is the Dirichlet average routing cost, and $\overline{Q} = \sum_{i \in \mathcal{V}} \overline{q_i}$ is the average network queue size, and $\beta \in [0, 1]$ is the control parameter to determine the trade-off between these two optimization goals. Throughput optimality for a routing algorithm refers to its ability to maintain all queues to be stable for all sets of arrival rates for which it is possible by an omniscient router to maintain stable queues. Network constraints include constraints on link rates as well as interference constraints.

The HD routing algorithm also has three steps, nonetheless, the details of these steps are significantly different from the BP routing algorithm. The steps can be described as follows.

3.2.1. HD Weighing

Calculate the number of packets the link will transmit if it is activated at time-slot n . In the original literature, this quantity is denoted by $\widehat{f_{ij}}(n)$, calculated as follows.

$$\begin{aligned} \widehat{f_{ij}}(n) &= \min\{\phi_{ij}(n)q_{ij}(n)^+, \mu_{ij}(n)\} \\ \phi_{ij}(n) &= (1 - \beta) + \beta/\rho_{ij}(n) \end{aligned} \tag{5}$$

where the Lagrange parameter β is defined in Eqn (4). Now, the link weights are calculated as follows.

$$w_{ij}(n) = 2\phi_{ij}(n)q_{ij}(n)\widehat{f_{ij}}(n) - \widehat{f_{ij}}(n)^2 \quad (6)$$

3.2.2. HD Scheduling

Similar to BP routing, find a scheduling vector $\pi \in \Pi$ such that

$$\phi(n) = \arg \max_{\pi \in \Pi} \sum_{ij \in \mathcal{E}} \pi_{ij} w_{ij}(n) \quad (7)$$

and the ties are broken randomly.

3.2.3. HD Forwarding

At this step, send $\widehat{f_{ij}}(n)$ number of packets over the link ij if $\pi_{ij}(n) = 1$ and $w_{ij}(n) > 0$. However $\widehat{f_{ij}}(n)$ may be a fractional number. Therefore the actual number of packets transmitted is

$$f_{ij}(n) = \begin{cases} \lceil \widehat{f_{ij}}(n) \rceil & \text{if } \pi_{ij}(n) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Note that, unlike BP routing, the node sends $f_{ij}(n)$ number of packets rather than transmitting at the full capacity, $\mu_{ij}(n)$.

3.3. The Backpressure Collection Protocol

Backpressure Collection Protocol (BCP) [2] is a distributed dynamic routing protocol which practically implements the idealized V-parameter BP algorithm without need for global max weight scheduling. In this protocol the link penalty $\theta_{ij}(n)$ in Eqn (2) is replaced by $\overline{ETX_{ij}}(n)$, which is the *ETX* estimate for link ij at time-slot n . Therefore, the modified weighing function is as follows.

$$w_{ij}(n) = \mu_{ij}(n)(q_{ij}(n) - V.\overline{ETX_{ij}}(n)) \quad (9)$$

In this distributed protocol, each node calculates the weight for each of its outgoing links locally and chooses the neighbor with the maximum positive

weight, if any, to forward the next packet. It is shown in [2] that a last-in-first-out (LIFO) queue implementation of BCP is better than first-in-first-out (FIFO) in terms of delay performance. Also, floating or virtual queues are implemented to deal with the problem of limited buffer size.

4. The Heat Diffusion Collection Protocol

The original HD algorithm is a centralized protocol where at each time slot the optimum non-interfering schedule must be computed. In our distributed implementation of the Heat Diffusion Collection Protocol (HDCP), every node decides the next hop locally, greedily based on the weight calculations. Moreover, the $\rho_{ij}(n)$ in Eqn (5) is replaced by $\overline{ETX}_{ij}(n)$ which is the estimated ETX of the link ij at time-slot n . Thus, the modified equations to calculate the link weights are as follows.

$$\begin{aligned}\widehat{f}_{ij}(n) &= \min\{\phi_{ij}(n)q_{ij}(n)^+, \mu_{ij}(n)\} \\ \phi_{ij}(n) &= (1 - \beta) + \beta/\overline{ETX}_{ij}(n)\end{aligned}\tag{10}$$

$$w_{ij}(n) = 2\{(1 - \beta) + \beta/\overline{ETX}_{ij}(n)\}q_{ij}(n)\widehat{f}_{ij}(n) - \widehat{f}_{ij}(n)^2\tag{11}$$

Now, each node calculates the weight for each of its outgoing links and chooses the link with the maximum positive weight. Unlike the centralized algorithm's NP-hard maximum weight independent set time scheduling to avoid interference, in our distributed protocol, any packet collisions that may occur are fixed by a random retransmission.

4.1. Updating Weights

In order to calculate the weights, the distributed HDCP protocol requires some techniques to ensure that each node have a updated information about the queue sizes of its neighboring nodes, without affecting the performance of the routing task.

In our distributed implementation of HDCP, we employ two techniques to do that. First, during a long period of inactivity, each node periodically broadcasts a beacon with its current queue status similar to common wireless access point. If a neighboring node receives this broadcast, it will update its locally stored queue differential information. Second, when a node sends a data packet, it includes its current queue state in that packet’s header. Due to the nature of wireless links, every packet is received by all the neighboring nodes (we assume that no advanced MAC protocol is employed that schedules nodes to communicate in pairs at different times). Once a data packet is received by a node, it sniffs the header of the packet to extract the queue information and updates the local queue information data-base, even if the respective node is not the destination of the packet.

4.2. Queue Implementation

Similar to BCP, practical implementation of the HDCP can have a FIFO queue or a LIFO queue implementation. Based on the observation in [2] that LIFO queue implementation has a significantly better performance in terms of end-to-end delay (which we also observed empirically), we present only the LIFO queue implementation of the HDCP protocol in this paper. We also adopt the virtual “floating” queue approach proposed in [2] to prevent packet buffer overflows due to the steady state queue gradient.

5. Implementation Details

In the practical implementation of HDCP, a number of parameters need to be set properly, such as β , maximum queue size, and maximum number of retransmissions. In this section, we discuss the choices of parameters and the reason behind them in details. First of all, we set the value of $\mu_{ij}(n)$ in Eqn. (5) to be 1 as a node cannot send more than one packet simultaneously. Next, we discuss the choice of β in details.

5.1. The β parameter

The most important parameter in HDCP is the parameter β . In theory, for different choices of β , we should get different performance for HDCP as the optimization goal changes for different values of β . If we choose $\beta = 0$, Eqn. (11) will be simplified to:

$$\begin{aligned}\widehat{f_{ij}}(n) &= \min\{q_{ij}(n)^+, 1\} \\ w_{ij}(n) &= 2q_{ij}(n)\widehat{f_{ij}}(n) - \widehat{f_{ij}}(n)^2\end{aligned}\tag{12}$$

Now, for $q_{ij}(n) > 0$, $\widehat{f_{ij}}(n) = 1$ as the queue differential can take only integer values. Therefore, Eqn (12) can be rewritten as follows:

$$w_{ij}(n) = \begin{cases} 2q_{ij}(n) - 1 & \text{if } q_{ij}(n) > 0 \\ 0 & \text{Otherwise} \end{cases}\tag{13}$$

Therefore, in this the optimization goal becomes similar to the goal in the original “pure” BP routing (by Tassiulas and Ephremides [1]) and it doesn’t include the minimization of ETX. Also, as the link weights solely depend on the queue differentials, the delay performance should be better *provided that the links are all very good*. On the contrary, since this protocol doesn’t try to minimize the ETX, it can choose a bad link if the queue gradient on that link is the largest. As a consequence, the average number of retransmissions faced by a packet also increases which is directly translated to larger end to end delay. Therefore, if the overall path costs in terms of ETX is the dominant factor in the end to end delay calculation, HDCP with $\beta = 0$ may perform poorly in practice.

On the other hand, if $\beta = 1$, Eqn. (11) will be as follows:

$$\begin{aligned}\widehat{f_{ij}}(n) &= \min\{\frac{q_{ij}(n)^+}{\overline{ETX_{ij}}(n)}, 1\} \\ w_{ij}(n) &= 2\frac{q_{ij}(n)}{\overline{ETX_{ij}}(n)}\widehat{f_{ij}}(n) - \widehat{f_{ij}}(n)^2\end{aligned}\tag{14}$$

Similar to the previous case, we can simplify (14) as follows:

$$w_{ij}(n) = \begin{cases} 2 \left(\frac{q_{ij}(n)}{\overline{ETX}_{ij}(n)} \right) - 1 & \text{if } \frac{q_{ij}(n)}{\overline{ETX}_{ij}(n)} \geq 1 \\ \left(\frac{q_{ij}(n)}{\overline{ETX}_{ij}(n)} \right)^2 & \text{if } 0 < \frac{q_{ij}(n)}{\overline{ETX}_{ij}(n)} < 1 \\ 0 & \text{Otherwise} \end{cases} \quad (15)$$

In this case the optimization goal is mainly the reduction of overall path costs in terms of ETX. Thus the overall ETX for a path should be improved for this case. However, this might result in a slight increase in hop counts if the links are very lossy. The first and last cases in Eqn. (15) correctly fulfill our routing requirement. But the second case causes inefficiency in the real testbed experiments. In such cases, even if the ETX cost is very high for a link and the queue differential is as low as 1, a node will try to send the packet to that link according to the original HD rule. Moreover, in practical experiments, the probability of falling under such a situation is very high. Thus, it will negatively effect the overall performance of the HDCP and needs to be avoided. Furthermore, provided that we have avoided any such situations, for a good link that have $ETX = 1$, even a queue differential of 1 will result in a positive weight thereby causing the protocol to forward the packet. This results in a absence of a steady state queue gradient on such links. This can potentially increase the number of hops traversed by the packets and also affect the goodput. In order to avoid both of these situations, we replace the $\rho_{ij}(n)$ in Eqn (5) by $\mathbf{V} \times \overline{ETX}_{ij}(n)$ which modifies Eqn. (11) as follows:

$$w_{ij}(n) = 2\{(1 - \beta) + \frac{\beta}{\mathbf{V} \times \overline{ETX}_{ij}(n)}\}q_{ij}(n)f_{ij}(n) - f_{ij}(n)^2 \quad (16)$$

where, $f_{ij}(n) = \lceil \widehat{f_{ij}}(n) \rceil$. By setting $\mathbf{V} \geq 2$, we make it certain that there exists a steady state queue gradient towards the sink. Therefore, a node will consider a link only if $q_{ij}(n)$ is greater than $\overline{ETX}_{ij}(n)$. Thus, for a link with very high ETX, the queue differential have to be higher in order to consider that link. Furthermore, this strategy also satisfies the Backpressure criterion as for $q_{ij}(n) < 0 \implies w_{ij}(n) < 0$. In Section 6.3, we present a practical experiment

based analysis of the performance improvement as a result of this change in weight calculation.

5.2. Retransmission

Retransmission is very crucial for the performance of any wireless network. For effective retransmission, the parameters such as retransmission timeout and maximum number of retransmissions have to be properly chosen. Retransmission is also directly related to the acknowledgement mechanism and the choice of ARQ. Since the choice of ARQ affects the HDCP, the BCP and the CTP algorithm equally, in this paper we have implemented a simple Stop and Wait ARQ mechanism where a node can send only one packet at a time and wait for its acknowledgement before moving to the next packet. If the acknowledgement is not received within a certain time, commonly referred as retransmission timeout, the node retransmits the same packet. Now the value of this retransmission timeout directly affects the goodput of the system and needs to be properly chosen. Note that, the ARQ mechanism is employed on top of the existing hardware level acknowledgement mechanism that tries a maximum of 3 times to properly transfer the packet to the next hop in case of unicast transmissions (e.g., software acknowledgements). We do not remove the hardware level acknowledgement (One key feature of the CTP algorithm) for a fair comparison as well as to avoid the unreliability issues in pure software acknowledgements.

In our experiment, the transmission and propagation time for a packet are in the order of tens of milliseconds. It would then perhaps be expected that the best setting for the retransmission should be on the order of around 10ms or so. Nevertheless, we empirically found that it is best to set the timeout for retransmitting a lost packet to be chosen randomly between 10 to 200 ms. We believe that this large range is needed because of the channel coherence time in our testbed which is located in a busy office building environment. For instance, in [22], it is indicated that the coherence time for IEEE 802.15.4 radios can be about 175ms. Retransmitting a lost packet quicker than the coherence time runs a higher risk of seeing another packet loss. Furthermore, we use CSMA as

the channel access protocol, which also introduces some delay.

The maximum number of retransmission attempts is set to 5 based on the original BCP code, which we empirically observed to perform well on our testbed. After five retransmission attempts, if a packet is not acknowledged, the node will drop it and move to the next packet.

5.3. *Retry*

Whenever a node generates or receives a packet, it tries to send it immediately (after about 4-5 ms) if no other packet is being transmitted or waiting in the queue. However, when the node wants to transmit the packet, there might not be any suitable neighbor (in terms of having a positive weight) to forward the packet. In that case, the node needs to decide how much time should it wait before retrying. We refer to this wait time as the Retry time. One viable option is to constantly keep trying which is not efficient in terms of energy consumption due to radio wake times. Also once this situation happens, it might take a while to have a good neighbor. In this work, we set the retry time to be chosen randomly between 50ms to 100ms. The intuition behind choosing this value is again the transmission time for a packet being in the order of tens of milliseconds. Based on our experiment, we have also observed that the typical packet transfer time (the time duration between the transmission and reception of a packet) is $\sim 10ms$. Therefore, by choosing a value between 50ms and 100ms, we give the neighboring nodes enough time to potentially transfer several packets which is likely to be enough to create a positive weight.

5.4. *Link Metric Estimation*

One of our contributions in this paper is to propose a new method of ETX calculation for implementations of dynamic routing. Initially, we opted to follow the ETX calculation technique from original BCP paper [2]. In that implementation the estimation of \overline{ETX}_{ij} for link ij is performed in an online manner where the metric is updated by taking exponential weighted moving average of the number of retransmission attempts of the most recently transmitted packet. This is a very effective way of ETX estimation for routing protocols that doesn't

switch next hop during retransmission i.e., use the same link ij for all the retransmission attempts. However, for Backpressure-based dynamic routing protocols, the next hop calculation is performed before each retransmission, for path diversity. In such cases, we have to be very careful in calculating moving average since the same link may not be used for all the retransmissions. Therefore, if we just attempt to update the ETX for the most recently used link with the total number of retransmission attempts, we found that it could potentially result in an erroneous ETX estimation. To avoid this flaw, we can keep track of all the links used as well as the number of tries on that link and update either only the last used link or all the links after a successful packet transmission or a packet drop.

As an alternative, we propose a 2-state discrete time Markov Chain based ETX estimation. In this method, we assume that each channel can be either good ('1') or bad ('0') at certain point of time. With each state, we associate two transition probabilities: good to good (p_{11}), good to bad (p_{10}), bad to good (p_{01}) and bad to bad (p_{00}). Now the \overline{ETX}_{ij} can be calculated as $\frac{1}{p_{01}}$ when the last state observed was a 0, and as $\frac{1}{p_{11}}$ when the last state was 1.

We maintain four counters associated with each routing table entry to keep track of different state transitions, denoted $count_{00}$, $count_{01}$, $count_{10}$ and $count_{11}$. We also add a Boolean variable to keep track of the last state of the link i.e., if the value is **true**, the last known state was good. We initialize the $count_{01}$ and $count_{11}$ to be 1 and the others to be 0. Now every time a packet is transmitted (or retransmitted), the algorithm waits for a certain period of time to receive the acknowledgement(ACK). If received, the state of the channel is set to be good ('1') otherwise it is set to be bad ('0'), and then based on the last state, the respective counter is increased by one. The counters may be reset after reaching a maximum value, to keep the ETX estimates fresh. In our experiments, we have not done this as it did not appear to affect the performance.

Now, based on the counters, the ETX is calculated (it can be shown that this corresponds to a maximum likelihood estimate of the underlying Markov

Chain parameters) as follows:

$$\overline{ETX}_{ij} = \begin{cases} \frac{count_{00}+count_{01}}{count_{01}} & \text{if last State} = 0 \\ \frac{count_{10}+count_{11}}{count_{11}} & \text{if last State} = 1 \end{cases} \quad (17)$$

All the results presented in this paper are based on this new, more justifiable method of ETX calculation, which we apply to both BCP and HDCP for a fair comparison.

5.5. Queue Buffer

In practical low power low memory devices, the possible queue buffer allocations are severely restricted. We fixed the maximum queue size to be 25 as this is the highest possible number of queue buffer that our device can accommodate alongside other required memories. Along with this buffer, there also exists a small memory allocated to store only the recent packet for the retransmission purpose.

5.6. Beacon Timer

Beaconing is a very important part of the practical implementation of both HDCP and BCP. When a node has nothing to send for long time, beacons are sent periodically, so that the neighboring nodes can keep their Backpressure database updated. Also beaconing is mandatory for a sink node since it has nothing to send. Therefore we implement two different beaconing rates in our system. The first type of beaconing is for source nodes and the period for that is around 5 seconds. The second type of beacons, which we refer to as the fast beacons, are used by the sink nodes and the period for that beacon is around 2 seconds. These values are chosen based on the original BCP code.

5.7. Inbound Packet Filtering

Inbound packet filtering is very important to improve the performance of both HDCP and BCP in the presence of retransmissions. If no filtering is used, a node might receive multiple copies of the same packet due to retransmissions. Therefore the node might have multiple copies of the same packet stored in the

buffer simultaneously, which is not efficient. To avoid this kind of situations, we implement a inbound packet filter to drop any duplicate packets after sending proper acknowledgements. In our implementation, each node maintains a history of 25 most recent packets received by the node. We choose this number to match the queue buffer size. Every time a node receives a packet, it checks the history, performs necessary action such as packet drop or store, and updates the history.

Further, to prevent packet looping, we implement a TTL counter which decrements at each hop. In our experiments, sources set the initial TTL for each packet conservatively to 10 (the maximum hop distance from any node to the sink in our testbed is only 3).

5.8. *Link Switching*

In this paper, we propose an enhancement of HDCP by introducing link switching. The main concept of link switching is to maintain a ordered set of best (in terms of the weights) K neighbors (K can be any positive integer) at each node. When a packet is sent, it is first send to the first neighbor on this list. If transmission fails, the retransmission attempt is made immediately to the next neighbor in the list and so on. If the list is exhausted during retransmissions, the process restarts again from the first neighbor in the list. A node should fulfill some selection criterion to be included in the list such as the link weight should be within some threshold of the best link. In our experiments, we set a threshold on the ETX and weight i.e., if a positively weighted link's ETX is no worse than the best link's $ETX + 1$, we add that link to the list. In section 6.3, we present a practical experiment based analysis of the performance improvement as a result of this change. However we introduce this switching in HDCP only because we empirically found that it doesn't help to improve the performance of BCP.

5.9. *End to End Delay Calculations*

For calculating end to end delay of each packet, we maintain a separate field called $HDCP_{Delay}$ in the HDCP header, initialized with a value of 0. At the source node, the packet is timestamped at the generation (Say A_{source})

and just before departure (Say D_{source}), and the value $HDCP_{Delay}$ field is set to be $(D_{source} - A_{source})$. Similarly, we time-stamp the packet at each intermediate node, I_k : upon arrival (A_{I_k}) and just before departure (D_{I_k}); and add the time difference with the value of $HDCP_{Delay}$, i.e., $HDCP_{Delay} = HDCP_{Delay} + (D_{I_k} - A_{I_k})$. Thus the value of the field $HDCP_{Delay}$ upon arrival on the sink denotes the end to end delay suffered by that packet. For illustration, assume that the travel path of a packet is $source \rightarrow I_1 \rightarrow \dots \rightarrow I_M \rightarrow sink$, A_{I_1}, \dots, A_{I_M} are the arrival times of the packet at the intermediate nodes, and D_{I_1}, \dots, D_{I_M} are the respective departure times. Then the end to end delay is $\sum_{i=1}^{i=M} |D_{I_i} - A_{I_i}| + |D_{Source} - A_{source}|$. Note that, we do not add the propagation delays as the value of propagation delays are negligible in our testbed setup.

6. Real Testbed Experiment Results and Analysis

In this section we evaluate a number of HDCP variants and also compare them with LIFO BCP with virtual queue implementation and CTP.

6.1. Experimental Setup

To analyze the performance of HDCP in a real sensor network and compare it with BCP and CTP, we have implemented the HDCP and the BCP algorithm both on Contiki OS and used the CTP implementation available with the Contiki OS. We perform a set of evaluation experiments on an indoor wireless network testbed called TutorNet [23] with forty five IEEE 802.15.4-base Tmote-sky nodes distributed over a floor with roughly 80,000 sq.ft of area. This testbed is also available for administered public use for approved research purposes including benchmarking protocols. The network topology is presented in Figure 1 where the marked node is the sink and the rest of the nodes are the source nodes and the furthest node is three hops away from the sink. We use the channel number 26 with Tmote sky power level 31 for this purpose. The number of neighbors to each node varies from 19 to 35 with an average of 29. Nonetheless, typically only about 7-8 of the neighbors are connected via good links ($ETX \approx 1$). Thus the topology is very diverse with a considerable number

of different paths between any two nodes in the network. On the negative side, a considerable amount of interference exists among the nodes, which limits the bandwidth. The data packets in our experiments are all 26 Bytes in size.



Figure 1: Real Experiment Testbed Topology

All the experiments are performed on weekdays during daytime with lots of moving people and physical objects around. Each experiment is performed for 35 min: the network settles down during the first 5 min and the data is collected during the next 30 min. Each experiment is repeated at least 10 times to improve the confidence levels. Note that, we discuss the experimental setup for low power stack in section 6.6.

We evaluate HDCP’s performance in terms of different values of β and different packet generation rates. We select the value of V to be 2 for both BCP and HDCP which has been empirically determined to be an efficient operating point for BCP in the original BCP paper (which we could also verify in our own experiments).

6.2. Variation of the β Parameter

We perform a set of practical experiments on the testbed with different values of β and packet generation rates of 1 packet per 4 seconds per source (i.e., 0.25 PPS) as well as 1 packet per 2 seconds (0.5PPS). The difference in performance is not prominent between the two source rates, thus we present the results only for the higher rate of 0.5 PPS in this section to keep the length of

the manuscript reasonable as well as to avoid presenting redundant information.

The goodput of each source node is defined to be the number of packets received by the sink from it over a one second interval. For visual clarity, all the plots presented in this section are sorted in terms of the goodputs of the individual nodes for the experiment with $\beta = 0$. The end to end delay calculation for each packet is performed by adding up all the queuing and processing delays in all intermediate nodes, as discussed in Section 5.9. This ignores the propagation times, which in any case are negligible compared to the processing delays.

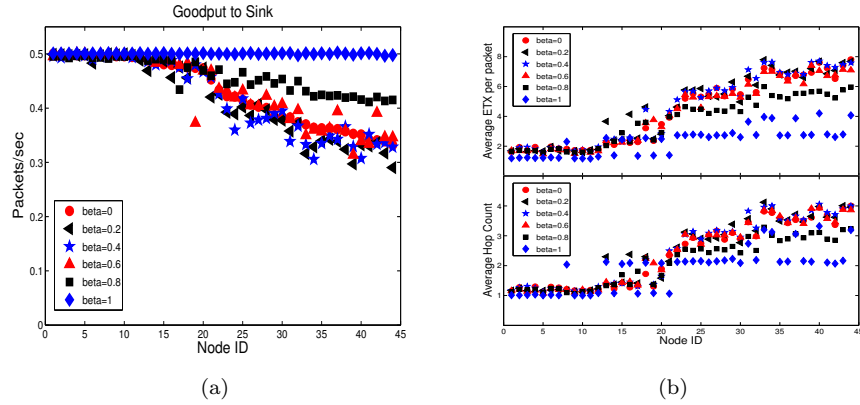


Figure 2: Performance Plots of HDCP Implementation for 0.5 PPS with Different Values of β : (a) Average Goodput to Sink (b) Average ETX per Packet (Top) and Average Hop Count (Bottom)

First, we analyze the goodput characteristics of HDCP for different choices of β . In Figure 2a, we compare the goodputs of each of the forty four nodes for six different choices of β . This figure clearly shows that the goodput for $\beta = 1$ and 0.8 are significantly better than the other choices of β . It also demonstrates that $\beta = 0$ results in a gradual decrease in the goodput to sink where only few nodes are able to reach the maximum possible rate. This figure also shows that the choice of $\beta \in \{0, 0.2, 0.4, 0.6\}$ does not significantly affect the goodput performance. Based on these observations, we hypothesize that the goodput

performance of the network is mostly dependent on the ETX of the path and therefore for higher β values the goodput performance metrics are better. Figure 2b, which shows that the average path costs for $\beta \in \{0.8, 1\}$ in terms of ETX for individual sources are significantly less than the average path costs for other choices of β , validates this hypothesis. In Figure 2b, we also analyze the average hop counts observed by the packets. It shows a similar pattern as the path costs since total ETX of the path is proportional to the number of hops traversed by the packet.

In Figure 3a (Bottom), we analyze the variation in the average end-to-end delay suffered by the packets generated from individual nodes for different values of β . It shows that the average delay performance for $\beta = 1$ is the best among different choices of β while any other choice of β results in a worse delay performance. Similar statistics are seen in Figure 3a (Top) in terms of the average queue sizes for individual nodes. This figure demonstrates that for $\beta = 1$ the average queue-sizes are almost three to four times smaller than that of the average queue sizes for $\beta \in \{0, 0.2, 0.4, 0.6\}$. We also plot the delay cdf in Figure 3b for the packets generated from mote 38 in the testbed which is the mote farthest from the sink. It also shows that $\beta = 1$ is best in terms of end-to-end delay.

Summarizing all these results, we can say that HDCP performs really well if the value of β is close to 1. For lower values of β , we find the performance does not differ by too much from the performance when $\beta = 0$ (the reason for this is further discussed in section 7.1). Therefore, we only consider HDCP with $\beta = 1$ and $\beta = 0$ (the latter as a baseline scheme, which does not take into account ETX) for the rest of the paper.

6.3. Modified HDCP vs Unmodified HDCP

In this section, we present a comparison of an HDCP implementation based on the original weighing model suggested by the theory with our HDCP implementation where the link weight model is modified as in (16) as well as with our proposed link switching approach. For this purpose, we perform a set of

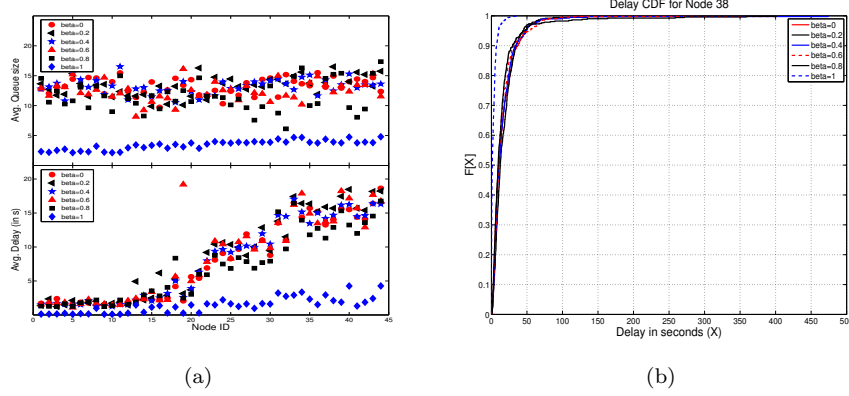


Figure 3: Performance Plots of HDCP Implementation for 0.5 PPS with Different Values of β : (a) (Bottom) Average End-to-End Delay, (Top) Average Queue Occupancy for Each Node (b) End-to-End Delay CDF Plot for Mote 38

experiments with both versions of HDCP for a fixed value of $\beta = 1$ and fixed packet generation rate of 0.25 PPS i.e., 1 packet per 4 seconds. Note that for visual clarity all the plots presented in this section are sorted in terms of the goodputs for unmodified HDCP implementation.

In Figure 4a, we demonstrate that without the modifications we have proposed, the goodput performance of HDCP suffers significantly. This is mostly due to the selection of links with higher ETX as well as lack of proper queue gradient towards the sink, as discussed in Section 5.1. This is further verified by the Figure 4b which clearly show that the average path costs in terms of ETX for unmodified HDCP are very high compared to our HDCP implementation.

Next, we compare the performance of unmodified and modified HDCP in terms of average end to end delay as well as average queue occupancy of individual nodes in Figures 5. It shows that the delay performance of unmodified HDCP is worse than modified HDCP for half of the nodes while it is better for the rest half of the nodes. Thus, on average, the modification doesn't attribute to any delay improvement. On the other hand, it is also clear from the figure that there exists a steady queue gradient in modified HDCP in contrary to the case of unmodified HDCP, where most of the nodes have queue occupancy of 1

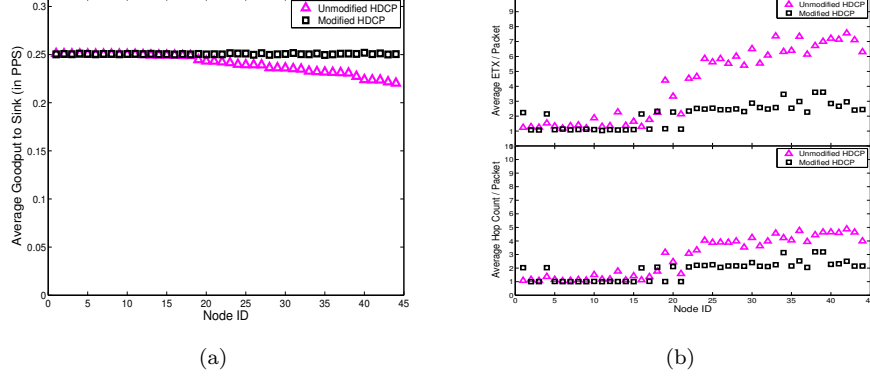


Figure 4: Performance Comparison between Modified and Unmodified HDCP Implementation with $\beta = 1$ for 0.25 PPS: (a) Average Goodput (b) Average ETX per Packet (Top) and Average Hop Count (Bottom)

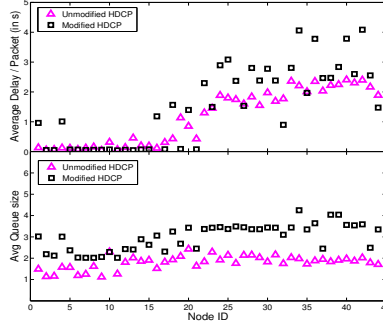


Figure 5: Average End-to-End Delay (Top) and Average Queue Occupancy (Bottom) Comparison between Modified HDCP and Unmodified HDCP with $\beta = 1$ for Each Node for 0.25 PPS

thereby lacking a proper queue gradient towards sink. This also validates our justification for the modification of weights in HDCP as indicated in Section 5.1. Thus, overall we improve the performance of HDCP by slightly compromising the average queue sizes.

6.4. Performance Comparison with BCP and CTP for Fixed Packet Generation Rate

In this section, we compare the performance of HDCP with the performance of the BCP protocol and the CTP protocol for the fixed packet generation rate of 0.5 PPS i.e., 1 packet per 2 seconds. Note that for simplicity of presentation, all the plots presented in this section are sorted in terms of the goodputs for the BCP algorithm.

In Figure 6a, we plot the goodputs for CTP, BCP and HDCP with $\beta = 0$ and 1, respectively. We observe that HDCP with $\beta = 1$ outperforms the CTP algorithm in terms of goodput while CTP outperforms HDCP for $\beta = 0$. However, BCP and HDCP with $\beta = 1$ performs almost identically. For $\beta = 0$, the weights of the links are fully determined by the queue differentials and it doesn't depend on ETX at all, resulting in bad performance. For CTP, a node relies on a single periodically calculated path to sink and doesn't take advantage of multiple available paths to sink thereby compromising the goodput for high packet generation rate such as 0.5 PPS. On the other hand, BCP and HDCP with $\beta = 1$ focus on reducing the total ETX cost of a source to sink path while not being restricted to a single pre-calculated path. Thus, the BCP and the HDCP algorithm with $\beta = 1$ both appear to be able to take advantage of the multiple paths available to the sink in order to cope with high packet generation rate thereby improving the throughput region.

Similar to the goodput analysis, we present the average hop count and average ETX of the entire path observed by the packets generated from individual sources in Figure 6b. It shows that, again, HDCP with $\beta = 1$ and BCP both slightly outperform CTP on average, whereas HDCP with $\beta = 0$ performs the worst. This is also justified based on our discussion presented in the previous section. The performance of HDCP with $\beta = 1$ and the performance of BCP are again almost same. **Based on these results, we hypothesize that the similarity between BCP and HDCP with $\beta = 1$ is due to similarity in their neighbor rankings, despite differences in the structure of the weight expression.** This is further explored in section 7.

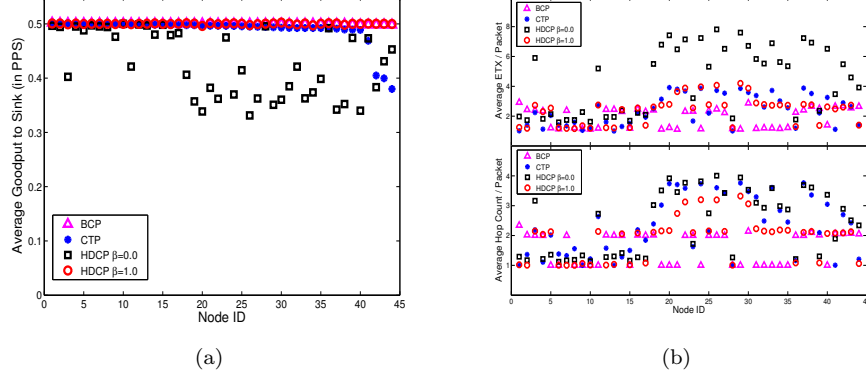


Figure 6: Comparison Plots between HDCP, BCP and CTP for 0.5 PPS: (a) Average Goodput to Sink (b) Average ETX (Top), Average Hop Count to Sink (Bottom)

We also compare the delay performance and queue occupancy of HDCP with BCP and CTP in Figure 7. Figure 7 shows that the delay performance of HDCP for $\beta = 1$ is significantly better than HDCP with $\beta = 0$. However, based on the figure, the delay performance for BCP is almost same as HDCP with $\beta = 1$ while both of them outperforms CTP. The similarity between BCP and HDCP with $\beta = 1$ is justified based on the previous results. In Figure 7, we also demonstrate that the average queue occupancy of HDCP with $\beta = 1$ is significantly low compared to BCP and HDCP with $\beta = 0$. The queue occupancy of CTP seems to be the lowest for some nodes, however, we believe this is misleading as CTP experiences the most packet drops among the various protocols at this offered load. The packet drops in CTP occur partly due to retransmission packet drops caused by higher intra-network interference (reflected in the higher ETX and higher delay values), and partly due to some other parameters in its implementation such as forwarding packet lifetime and an in-built congestion control. However, for any higher packet generation rate, we observe that the queue occupancy for CTP increases rapidly (resulting in even more losses) as does its delay.

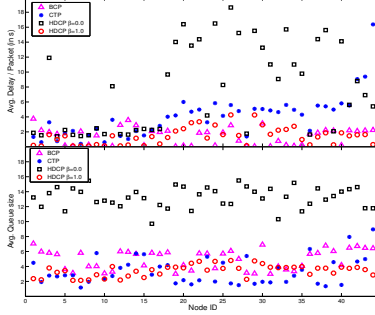


Figure 7: Average End-to-End Delay (Top) and Average Queue Occupancy Comparison (Bottom) among HDCP, BCP and CTP for Each Node for 0.5 PPS

6.5. Varying Packet Generation Rate

In this section, we present and analyze the effects of the packet generation/source rates on the performance of HDCP and compare it with the performance of the BCP and CTP algorithms. We performed a set of experiments with six different packets generation rates: 1/12 PPS (i.e., 1 packet per 12 second), 1/8 PPS, 1/4 PPS, 1/2 PPS, 4/5 PPS and 1 PPS. In Figure 8 we present the goodput variation due to the change in packet generation rate for HDCP with $\beta = 0$ and 1 as well as the goodput variations of the BCP and the CTP algorithm. It is clear from Figure 8 that for lower packet generation/source rates, HDCP performs almost similar to the BCP and CTP algorithm in terms of goodput to sink. But, as we increase the offered load, HDCP and BCP gradually outperform the CTP algorithm. In our experiment, HDCP outperforms CTP in terms of goodput for packet generation rate higher than 1 packet per 4 seconds. From the figure, we can estimate that the full throughput region (the maximum offered load at which the protocol is able to match the ideal curve) for HDCP is about 60 to 100% higher than that for CTP in this particular testbed and topology (of course the relative performance improvement is certainly likely to depend on the network topology.) Another thing to notice that, the average goodput for $\beta = 1$ is always higher than $\beta = 0$ which agrees

with our earlier findings and arguments concerning the inefficiencies introduced by ignoring the ETX costs of links. Yet again, the performance of BCP closely follows the performance of HDCP with $\beta = 1$ which is, again, due to similarity in their neighbor rankings in terms of the weights. This is further explained in section 7.

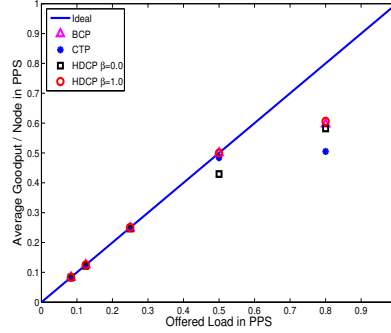


Figure 8: Variation of Goodput for Varying Offered Load

Next, we investigate the effects of increasing packet generation rates on the average path costs in terms of ETX and the average number of hops traversed by the packets. We plot the average ETX and average hop counts due to different source rates for HDCP, BCP and CTP in Figure 9a. It is observable from the figure that for any packet generation rate overall path cost for HDCP with $\beta = 1$ is comparable to BCP while CTP outperforms both for packet generation rate lower than 0.25PPS and converges with them for higher rates. Moreover, the average path cost for HDCP with $\beta = 0$ is higher than $\beta = 1$ which is justified by our discussion in the previous section. Similar statistics is available from the plot of average numbers of hops encountered by each packet due to its direct relation with the overall path ETX. The similarity between HDCP with $\beta = 1$ and BCP is, again, justified based on our earlier discussions. The apparent ‘good’ performance of CTP is due to its increasing incapability of sending packets with long path costs to the sink as it encounters congestion drops.

Lastly, we analyze the effect of packet generation rate on the average delay in

Figure 9b. Although CTP and BCP outperforms HDCP for source rates lower than 0.25PPS by a small margin, this figure demonstrate the superiority of the HDCP for $\beta = 1$ in overall delay performance as it continue to guarantee lower delay for higher packet generation rates. Another interesting fact to notice is that for BCP and HDCP, the delay gradually increases with packet generation rate whereas the delay for CTP increases rapidly with packet generation rate. This is likely because BCP and HDCP can take advantage of multiple paths to sink whereas the CTP relies on only one path. Therefore, CTP reaches congestion earlier than BCP and HDCP, which results in the rapid increase in delay. Again, the similarity between BCP and HDCP with $\beta = 1$ is due to similarity in the neighbor ranking in terms of the weights.

To summarize, our experiments lead us to conclude that optimized combinations of queue-awareness and ETX (implemented in BCP and HDCP with $\beta = 1$) provide the best choice for routing, better than routing based on ETX alone (CTP), which in turn, performs better than queue-aware routing alone (HDCP with $\beta = 0$).

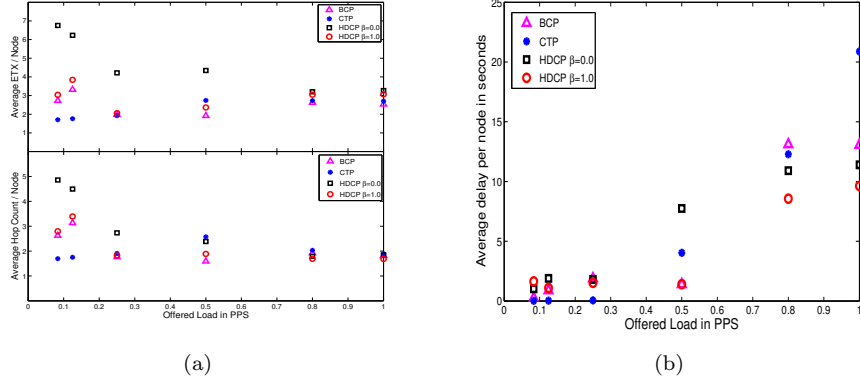


Figure 9: (a) Variation of Average Path Cost in Terms of ETX (Top) and Average Hop Count (Bottom) for Varying Offered Load (b) Variation of Average End to End Delay for Varying Offered Load

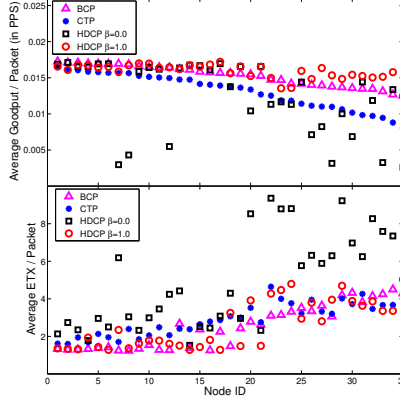


Figure 10: Performance Comparison of HDCP with BCP and CTP for a Low Power Communication Stack: (Top) Goodput to Sink, (Bottom) Average ETX Path Costs to Sink

6.6. Low Power Communication Stack Based Experiments

In order to verify the performance of HDCP on a low power communication stack, we performed a set of experiments with 35 sources and a sink (first 36 nodes of the testbed). For these experiments, we used CX-MAC protocol, a version of X-MAC[7] that is provided in Contiki, with duty cycle of 5% for HDCP, BCP and CTP. However, the choice of CX-MAC protocol over the other protocols is just a matter of the availability of Contiki implementation. Furthermore, since we are using a duty cycle, we also need to cut-back our source rates to a very low rate. For the presented set of experiments, we used a packet generation rate of 1 packet per 60 seconds (i.e., 1/60 PPS). We present the results in Figure 10. This figure shows that the HDCP protocol with $\beta = 1$ performs well in a low power communication stack, at a very low duty cycle setting where even CTP shows some deterioration in fairness of goodput. However, in this setting the performance with the baseline of $\beta = 0$ is much worse, leading us to conclude that it is a very poor setting indeed. Now, in order to estimate the actual energy consumptions, we record the different energy consumption components using the Contiki PowerTrace tool (in terms of the percentage of time

spent in different radio phases: Transmit, Listen/receive). Based on our traces, in HDCP with 5% duty cycle, the radio of each node is on for 5.92% of the total execution time, out of which the node is transmitting and receiving approximately 0.65% and 5.27% of the total execution time, respectively. Now to get the actual energy consumption, one can use the current and voltage ratings from the specifications of the devices used. For example, in Tmote-sky the rated voltage of operation is approx 3.3V and the average current consumptions are 17.4mA and 19.7mA for radio transmission and radio reception, respectively. This results in approximately 113.78mJ energy consumption in each Tmote-Sky for the experiment period of 30 minutes.

6.7. External Interference

In this section, we evaluate the performance of the HDCP protocol with the optimized $\beta = 1$ in the presence of external interference and compare it with both BCP and CTP. This is necessary because the 802.15.4 radios share frequency band with WiFi, Bluetooth, and other Zigbee radios and as a result their performance often suffers from severe interference. To emulate such scenarios, we performed a set of experiments with forty sources and a single sink (Node 1) while four nodes are used as interference sources on channel 26. The interfering nodes are inactive for the first five minutes of the experiment, periodically transmit for next fifteen minutes and become inactive again for the last five minutes of the experiment. During the on period, each of the interfering nodes transmits 110 Byte packets at a rate of 100PPS for 15 seconds and then does not transmit anything for the next 15 seconds, and so on. Furthermore, we reduced the power level of all 41 nodes from level 31 to level 15 whereas the interfering nodes were kept at level 31, in order to intensify the effect of interference. The outcome of this set of experiments is presented in Figure 11a that plots the delivery percentage of the packets over a series of 30 seconds time window for HDCP with $\beta = 1$, BCP and CTP. It demonstrates that while CTP performance significantly suffers from the interference, the HDCP protocol maintains its good packet delivery ratio, similar to BCP.

The above mentioned settings is used to stay consistent with the interference settings presented in the original BCP paper[2]. However, it is well known that the simple Gilbert-Eliot model used for ETX estimation might work perfectly with some specific synthetic interference models and might fail in realistic interference scenarios. In order to explore the performance of the HDCP algorithm, in presence of real interference, we perform a set of experiments with 44 source nodes and 1 sink node, running on channel 13 of the 802.15.4 standard which is known to be one of the most interfered channels. We also compare the performance of HDCP based on the Gilbert-Eliot (GE) ETX model with the performance of BCP with the GE model as well as with HDCP based on the ETX model used in the original BCP paper[2]. For this set of experiments, we do not use the link switching method as we empirically found that interference itself causes sufficient link switching, thereby, adding extra link switching negatively affects performance. The results presented in Figure 11b clearly demonstrate that even in the presence of constant real interference, the HDCP algorithm with Gilbert-Eliot ETX model performs comparable to the BCP algorithm and the HDCP algorithm with the basic ETX model presented in [2], while outperforms the CTP algorithm. Furthermore, both Figures 11a and 11b show that the BCP and the HDCP algorithm can achieve approx 85% delivery ratio in presence of interference while the CTP achieves approx 70%.

7. Similarity Analysis Between HDCP and BCP

In this section, we analyze the BCP and the HDCP algorithm to identify the reasons behind the similarity in their performance. The performance of both the BCP and the HDCP depend on the rankings of the neighbors (based on the link weighing functions) of a node, which in turn translates to selection of routing paths to sink. From theoretical standpoint, the performance of HDCP and BCP will be different in a network if their respective rankings of the neighbors under same queue occupancy conditions are different. **Conversely, we hypothesize that the similar rankings of neighbors for both the HDCP and the BCP protocol will result in similar performance.**

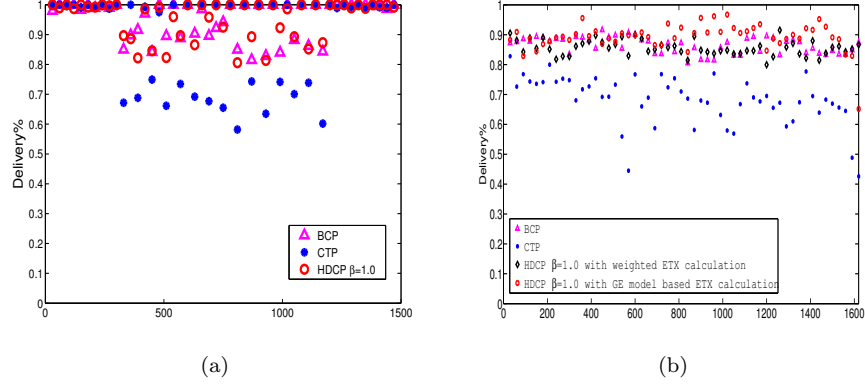


Figure 11: Thirty Second Windowed Average Sourced Packet Delivery Ratio for: (a) Synthetically Generated Interfering 802.15.4 Channel 26 Traffic (b) Real Interference Scenario on 802.15.4 Channel 13

In order to analyze the scenarios that will result in different or similar rankings of neighbors for HDCP and BCP, we compare the simplified weighing functions of BCP and HDCP with $\beta = 1$, which can be written as follows:

$$\begin{aligned} w_{ij}^{bcp}(n) &= q_{ij}(n) - 2 \cdot \overline{ETX}_{ij}(n) \\ w_{ij}^{hdcp}(n) &= \frac{q_{ij}(n) - \overline{ETX}_{ij}(n)}{\overline{ETX}_{ij}(n)} \end{aligned} \quad (18)$$

provided that $V = 2$ and $\frac{q_{ij}(n)}{2\overline{ETX}_{ij}(n)} \geq 1$, i.e., the links have non zero weights according to both BCP and HDCP weighing schemes. First of all, we try to identify the range of the possible network configurations that will result in different rankings for HDCP and BCP. For this purpose, we analyze a toy topology illustrated in Figure 12. Assume that the ETX_{31} and ETX_{32} are 1 and $e \geq 1$, respectively. Now, the weights of the respective links according to BCP will be:

$$w_{31}^{BCP} = q_{31} - 2, \quad w_{32}^{BCP} = q_{32} - 2e \quad (19)$$

Similarly, the weights for the links according to the HDCP rule for $\beta = 1$ will be (provided that $q_{31} \geq 2$ and $q_{32} \geq 2e$):

$$w_{31}^{HDCP} = q_{31} - 1, \quad w_{32}^{HDCP} = \frac{q_{32}}{e} - 1 \quad (20)$$

Now,

$$\begin{aligned} w_{31}^{BCP} &> w_{32}^{BCP} \quad \text{if } e > (q_{32} - q_{31})/2 + 1 \\ w_{31}^{HDCP} &> w_{32}^{HDCP} \quad \text{if } e > \frac{q_{32}}{q_{31}} \end{aligned} \quad (21)$$

Thus, if $\frac{q_{32}}{q_{31}} < e < (q_{32} - q_{31})/2 + 1$ the rankings of the outgoing links of node 3 are different, while the rankings are the same for all other values of e . As an example, say, $q_{31} = 4$ and $q_{32} = 6$, then only for $3/2 < e < 2$, the rankings are different. However, according to Eqn. (18) as well as Eqn. (21), for $ETX = 1$ both schemes will put similar weights on the links but with different negative offsets (2 for BCP and 1 for HDCP). Thus, the steady state performance will be same for both but with slightly lesser queue sizes in HDCP, which is also verified by our experiments. To verify whether the presence of too many perfect links is one of the reason behind the similar performance of HDCP and BCP, we plot the CDF of the ETX traces collected from all the nodes during a real collection experiment, in Figure 13a. In Figure 13b, we plot the CDF of the average link costs (average ETX per link) of the shortest paths between every possible pairs of nodes in the testbed. Figure 13a illustrates that a significant number ($\approx 40\%$) of links are perfect links ($ETX \approx 1$) while Figure 13b implies that approximate 40% of the shortest paths consists of only perfect links ($ETX \approx 1$). Furthermore, approximate 60% of the shortest paths in the network between any possible node pair consists of links with average ETX of 1.25, as shown in Figure 13b. All these statistics suggest similarity in the rankings of neighbors as well as similarity in performance for the BCP and the HDCP algorithms. In summary, since we do not observe much of a difference in the performance of the HDCP and the BCP algorithms, we conjecture that in our experiment setup, the probabilities for different rankings of the neighbors (more specifically, top 2 neighbors) are very low.

Next, in order to verify whether similarity in neighbors' ranking will result in similar performance, we perform a theoretical analysis of the steady state queue gradients for both HDCP and BCP. The steady state queue sizes depends on the smallest cost path to the sink. Say, for a node i , there ex-

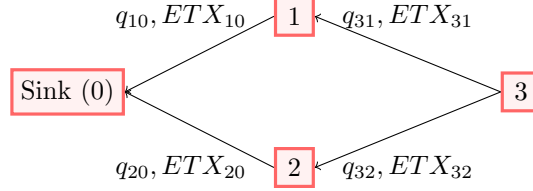


Figure 12: A Simple Topology For Ranking Similarity Analysis Between HDCP and BCP

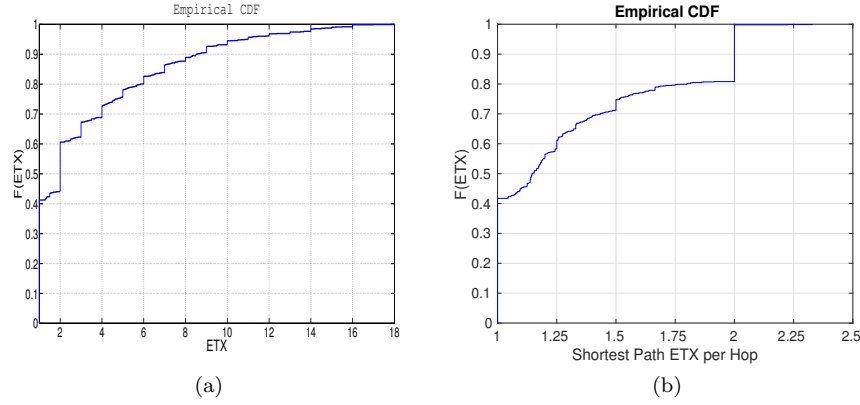


Figure 13: (a) Empirical CDF of the Link ETX Values for Our Testbed (b) Empirical CDF of the Average ETX per Link for the Shortest Paths Between Any Pair of Nodes

ists $k \in \{1, 2, \dots, K\}$ possible paths and each path consists of one or more links l_k . Then the steady state queue size for node i in BCP will be $\bar{w}_i^{bcp} = \min_{k \in \{1, 2, \dots, K\}} \left[\sum_{j=1}^{l_k} 2 * ETX_{k,j} \right]$, while in HDCP the steady state queue size will be $\bar{w}_i^{hdcp} = \min_{k \in \{1, 2, \dots, K\}} \left[\sum_{j=1}^{l_k} ETX_{k,j} \right]$, where $ETX_{k,j}$ represents the ETX of the j^{th} link of the k^{th} path from node i to the sink. Thus, we can say that the steady state path to sink for each node in HDCP is same as the BCP. Now, if the packet generation rate is low, every packet will always follow the steady state gradient and, thereby, follow the same path leading to similar performance. Now, if the packet generation rate is high, in worst case we will have a batch arrival of packets at some node, say i . Let us assume that when node i disseminate the batch arrival packets, all the neighboring nodes of i are

unchanged, i.e., no packet arrival (except from the node i) or departure takes place. In this situation, node i will keep on transmitting to the neighbor that is part of the best path to sink, until a point when the weight for the respective link becomes worse than the 2nd best link. In the following, we analyze at what point, i.e., after how many packet transmissions, node i will switch to the second best link.

- **BCP:** Node i prefers a neighbor node m over another neighbor node n , iff:

$$q_i - q_m - 2 \times etx_{im} > q_i - q_n - 2 \times etx_{in} \quad (22)$$

where q_i, q_m, q_n represent the queue sizes at node i, m , and n , respectively and etx_{im}, etx_{in} represents the etx of the links im and in , respectively. Now, say after x number of transmissions, the 2nd link is considered.

$$\begin{aligned} \implies (q_i - x) - (q_m + x) - 2 \times etx_{im} &= (q_i - x) - q_n - 2 \times etx_{in} \\ \implies x &= q_n - q_m + 2 \times (etx_{in} - etx_{im}) \end{aligned} \quad (23)$$

Now, WLOG assume that the node m is part of the best path to the sink from node i , while node n is part of the second best path. Then, $etx_{im} \approx etx_{in} \implies x \approx q_n - q_m$. If $etx_{im} < etx_{in}$, $x = q_n - q_m + 2 \times (etx_{in} - etx_{im}) > q_n - q_m$. On the other hand if $etx_{im} > etx_{in}$ implies $(q_m < q_n)$ for feasibility of the rankings in focus, which implies $x = q_n - q_m - 2 \times (etx_{im} - etx_{in}) \leq (q_n - q_m)$.

- **HDCP:** Node i prefers a neighbor node m over another neighbor node n , iff:

$$\frac{q_i - q_m}{etx_{im}} > \frac{q_i - q_n}{etx_{in}} \quad (24)$$

where q_i, q_m, q_n represent the queue sizes at node i, m , and n , respectively and etx_{im}, etx_{in} represents the etx of the links im and in , respectively.

Now, say after x number of transmissions, the 2nd link is considered.

$$\begin{aligned}
&\Rightarrow \frac{(q_i - x) - (q_m + x)}{etx_{im}} = \frac{(q_i - x) - q_n}{etx_{in}} \\
&\Rightarrow x(2 - \frac{etx_{im}}{etx_{in}}) = q_i \times (1 - \frac{etx_{im}}{etx_{in}}) - q_m + q_n \times \frac{etx_{im}}{etx_{in}} \\
&\Rightarrow x = q_i \times \frac{(1 - \frac{etx_{im}}{etx_{in}})}{(2 - \frac{etx_{im}}{etx_{in}})} - q_m \times \frac{(1 - \frac{etx_{im}}{etx_{in}})}{(2 - \frac{etx_{im}}{etx_{in}})} - q_m \times \frac{(\frac{etx_{im}}{etx_{in}})}{(2 - \frac{etx_{im}}{etx_{in}})} \\
&\quad + q_n \times \frac{\frac{etx_{im}}{etx_{in}}}{(2 - \frac{etx_{im}}{etx_{in}})} \\
&\Rightarrow x = \frac{1}{2} \times (1 - \mathbf{z}) \times (q_i - q_m) + \mathbf{z} \times (q_n - q_m) \text{ where } \mathbf{z} = \frac{\frac{etx_{im}}{etx_{in}}}{(2 - \frac{etx_{im}}{etx_{in}})} \\
&\hspace{15em} (25)
\end{aligned}$$

Now, WLOG assume that the node m is part of the best path to the sink while node n is the second best path. Similar to BCP, $etx_{im} \approx etx_{in} \Rightarrow x \approx q_n - q_m$. It also suggest that in HDCP, the number of transmissions before switching depends on a weighted sum of the queue differential of the best link $(q_i - q_m)$ and the queue differential of the 2nd best and the best neighbor $(q_n - q_m)$, where the weights depend on the ratio $(\frac{etx_{im}}{etx_{in}})$. If $etx_{im} < etx_{in}$, $0.5 \leq \mathbf{z} \leq 1$ that implies more weight on $(q_n - q_m)$ thereby increasing the chances of switching as $q_i \geq \max\{q_m, q_n\}$ which also implies $x \geq \frac{1+\mathbf{z}}{2}(q_n - q_m) \geq (q_n - q_m)$. On the other hand, $etx_{im} > etx_{in} \Rightarrow (q_m < q_n)$ for feasibility of the rankings in focus and $\mathbf{z} > 1$ that suggests $x = \mathbf{z} \times (q_n - q_m) - \frac{\mathbf{z}-1}{2} \times (q_i - q_m) \leq \frac{1+\mathbf{z}}{2}(q_n - q_m)$.

The above analysis suggests that if the outgoing best and 2nd best link of a node have similar etx, both BCP and HDCP will switch after exactly same number of transmissions, under same queue conditions. Even in other cases for any particular network, the switching patterns are similar and just switches after slightly different number of transmissions, which is a function of $(q_n - q_m)$. Therefore, the observed performance of BCP and HDCP will be similar. However, this analysis is pertinent to the fact that both HDCP and BCP have same rankings of the neighbors (atleast best two neighbors) which validates our hypothesis.

Based on the theoretical analysis, we conjecture that the similarity of performance between HDCP and BCP in our testbed experiments is due to similarity of rankings of the neighbors in most of the nodes. To verify this conjecture, we perform a Kendall Tau test of the ranking data collected from our real experiment setup, as follows.

7.1. *Kendall's Tau Test*

In the previous sections, we observed that the optimized versions of BCP and HDCP with $\beta = 1$ are very similar to each other in performance. We hypothesized that this may be due to similarity in the neighbor rankings for the two protocols. In order to verify our hypothesis, we collected a set of routing table snapshots from three representative nodes, located at a one hop, two and three hop distance from the sink, respectively, during a real collection experiment. These snapshots contain the information about their neighbors such as backpressure and ETX information from real experiment. Based on those snapshot values, we calculated the Kendall's Tau distance between the neighbor rankings generated by the weight calculation in BCP on one hand, and the neighbor rankings generated by the weight calculation in HDCP for different values of β on the other, for all neighbors that have a positive weight in at least one of the two protocols under comparison. Kendall's Tau distance between two rankings indicates the fraction of pairs that are ordered the same in the two rankings. If it is 0, then the two rankings are identical. Higher values indicate more different rankings.

We present the results in Figure 14. It clearly shows that while there is a lack of correlation for lower values of β , for $\beta \rightarrow 1$ there is a strong correlation between HDCP and BCP. This verifies our hypothesis and justifies the results shown in this paper.

Another noticeable fact is that for $\beta \in [0, 0.6]$ the Kendall Tau distance with respect to BCP remains almost the same. We performed additional Kendall's Tau correlation analysis between neighbor rankings of HDCP for every possible pair of $\beta \in \{0, 0.2, 0.4, 0.6\}$ and the average distance for each case was found

to be less than 0.1. This is the reason behind the similarity in performance of HDCP with $\beta \in \{0, 0.2, 0.4, 0.6\}$.

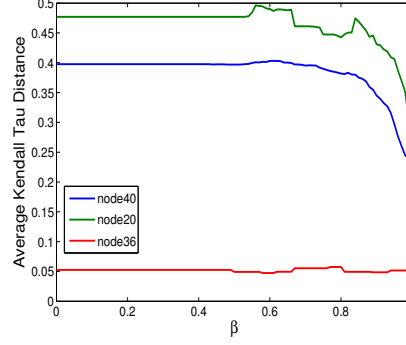


Figure 14: Variation of Kendall's Tau Distance between HDCP and BCP Neighbor Rankings for Different Values of β

8. Conclusion

We have proposed and implemented a new collection protocol for wireless sensor networks called HDCP that is the first practical realization of a theoretical algorithm inspired by Thermodynamics. We have evaluated HDCP on a real 45-node wireless sensor network testbed. We have compared the performance of HDCP with two well-known protocols CTP and BCP on this testbed. Based on the results, we can conclude that HDCP with an optimized parameter setting of $\beta = 1$ performs as well as BCP and outperforms CTP with respect to throughput performance, interference resilience, and low power operation, while all three generally offer about the same end-to-end delay on average in the full throughput region.

The equivalent performance of HDCP to the previously published BCP is a somewhat surprising finding of this study. From a mathematical perspective, this is not obvious as they employ quite different equations for the weight calculations and indeed in our prior theoretical works Heat Diffusion has been found to perform better than Backpressure scheduling in some respects. But as we

have shown, nevertheless, the two protocol implementations provide very similar neighbor rankings in a real network. We believe our finding also lends some support to the notion that it may not be possible to get any higher performance in practice with a dynamic routing protocol that takes into account both queue states and link quality.

The relative performance of BCP and HDCP in the presence of node mobility is of interest to evaluate in future work, as are extensions of HDCP that can work with IP packets. We would also like to understand how to optimize the various link transmission attempt timers from a more theoretically-informed perspective.

References

- [1] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.
- [2] Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, and Omprakash Gnawali. Routing without routes: The backpressure collection protocol. In *Proceedings of the IEEE IPSN 2010*.
- [3] Reza Banirazi, Edmond Jonckheere, and Bhaskar Krishnamachari. Heat-diffusion: Pareto optimal dynamic routing for time-varying wireless networks. In *Proceedings of the IEEE INFOCOM 2014*.
- [4] Reza Banirazi, Edmond Jonckheere, and Bhaskar Krishnamachari. Dirichlet’s principle on multiclass multihop wireless networks: minimum cost routing subject to stability. In *Proceedings of the ACM MSWiM 2014*.
- [5] Adam Dunkels, Oliver Schmidt, Niclas Finne, Joakim Eriksson, Fredrik Österlind, Nicolas Tsiftes, and Mathilde Durvy. The contiki os: The operating system for the internet of things, 2011.

- [6] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the ACM ENSS 2009*.
- [7] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the ACM SenSys 2006*.
- [8] JG Dai and Wuqin Lin. Asymptotic optimality of maximum pressure policies in stochastic processing networks. *The Annals of Applied Probability*, 18(6):2239–2299, 2008.
- [9] Devavrat Shah and Damon Wischik. Optimal scheduling algorithms for input-queued switches. In *Proceeding of IEEE INFOCOM 2006*.
- [10] Mohammad Naghshvar, Hairuo Zhuang, and Tara Javidi. A general class of throughput optimal routing policies in multi-hop wireless networks. *IEEE Transactions on Information Theory*, 58(4):2175–2193, 2012.
- [11] Jose Nunez-Martinez, Josep Mangles-Bafalluy, and Marc Portoles-Comeras. Studying practical any-to-any backpressure routing in wi-fi mesh networks from a lyapunov optimization perspective.
- [12] Majed Alresaini, Maheswaran Sathiamoorthy, Bhaskar Krishnamachari, and Michael J Neely. Backpressure with adaptive redundancy (bwar). In *Proceedings of the IEEE INFOCOM 2012*.
- [13] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. Efficient network flooding and time synchronization with glossy. In *Proceedings of the IEEE IPSN 2011*.
- [14] Nicolas Burri, Pascal Von Rickenbach, and Roger Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proceedings of the IEEE IPSN 2007*.

- [15] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. Low-power wireless bus. In *Proceedings of the ACM SenSys 2012*.
- [16] Olaf Landsiedel, Euhanna Ghadimi, Simon Duquennoy, and Mikael Johansson. Low power, low delay: opportunistic routing meets duty cycling. In *Proceedings of the IEEE IPSN 2012*.
- [17] Mobashir Mohammad, XiangFa Guo, and Mun Choon Chan. Oppcast: Exploiting spatial and channel diversity for robust data collection in urban environments. In *Proceedings of the IEEE IPSN 2016*.
- [18] T. Winter, P. Thubert, and RPL Author Team. Rpl: Ipv6 routing protocol for low-power and lossy networks, ietf rfc 6550. March 2012.
- [19] Srikanth Nori, Suvil Deora, and Bhaskar Krishnamachari. Backip: Back-pressure routing in ipv6-based wireless sensor networks. usc ceng technical report ceng-2014-01.
- [20] Leonidas Georgiadis, Michael J Neely, and Leandros Tassiulas. *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.
- [21] Michael J Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211, 2010.
- [22] Shahin Farahani. *ZigBee wireless networks and transceivers*. Newnes, 2011.
- [23] TutorNet. <http://anrg.usc.edu/www/tutornet/>.